Generación de librerías de código base para autenticación a través de certificados SSL generados automáticamente utilizando Java

Javier Vargas[1], David Guevara[1], Franklin Mayorga[1], Franklin Sánchez[2], Daniel Díaz[3]

[1] Facultad de Ingeniería en Sistemas, Electrónica e Industrial, Universidad Técnica de Ambato, Av. Los Chasquis y Río Payamino, Ambato, Ecuador.

[2] Departamento de Electrónica, Telecomunicaciones y Redes de Información, Escuela Politécnica Nacional, Ladrón de Guevara E11-253, Quito, Ecuador.

[3] Carrera de Ingeniería de Sistemas, Universidad Politécnica Salesiana, Av. Rumichaca S/N y Av. Móran Valverde, Quito, Ecuador.

Autores para correspondencia: jvargas1679@uta.edu.ec, dguevara@uta.edu.ec, fmayorga@uta.edu.ec, franklin.sanchez@epn.edu.ec, ddiaz@ups.edu.ec.

Fecha de recepción: 19 de junio del 2016 - Fecha de aceptación: 24 de julio del 2016

ABSTRACT

This paper describes the processes involved in the creation and evaluation of a Tool for Automatic Generation of Security Infrastructure in Communications using Java. Incorporation of security services in the source code of distributed applications is not that easy and requires, on the one hand the creation of a manual for the security of infrastructure based on digital certificates of public and private OpenSSL keys, and/or security stores. On the other hand, the application developer must incorporate manually in the source code the functions and/or procedures that properly manage certificates, keys and stores, to secure automatic implementation of the security services.

Keywords: OpenSSL, Automatic Generation, digital certificates, security stores.

RESUMEN

El presente trabajo describe el proceso realizado y resultados obtenidos de la creación de una Herramienta para la Generación Automática de Infraestructura de Seguridad en Comunicaciones usando Java. La incorporación de estos servicios de seguridad en el código fuente de las aplicaciones distribuidas no es fácil, se requiere, por un lado, la creación manual de una infraestructura de seguridad basada en certificados digitales de clave pública y privadas OpenSSL, y/o almacenes de seguridad. Por otro lado, el desarrollador de aplicaciones debe de incorporar, también manualmente, en su código fuente aquellas funciones y/o procedimientos que gestionen de forma adecuada los certificados, claves y almacenes con el objetivo de implementar los servicios de seguridad de forma automatizada.

Palabras clave: OpenSSL, Generación Automática, certificados digitales, almacenes de seguridad.

1. INTRODUCCIÓN

El amplio uso de Internet para el envío y recepción de información considerable, contempla la gestión de sistemas de seguridad en la comunicación entre un emisor y receptor representado en el concepto de cliente/servidor. Los sistemas tales como, software de comunicación electrónica, aplicaciones de web segura, conexiones a terminales, etc., requieren la implementación de servicios de seguridad, integridad, confiabilidad al instante de negociar la conexión entre cliente/servidor, para así salvaguardar su información. Actualmente existen herramientas y métodos para la seguridad de la información, una herramienta para la utilidad de gestión de claves públicas y privadas es IKEYMAN de IBM (Park, 2007), un método de autenticación segura y cifrado de aplicaciones web es SSL, TLS y HTTPS (Clark, 2013), con el propósito de mantener la seguridad de la información aplicando métodos

de encriptación como por ejemplo el algoritmo RSA (Gröbert, 2010), referencial a OpenSSL (Khalil-Hani, 2010), para la obtención de la clave de acceso tanto para el cliente como para el servidor, así estas entidades necesitan contar con una autoridad de certificación CA (López Jiménez, 2015), capaz de generar claves públicas TrustStore y privadas KeyStore (Lakhe, 2014), almacenes de seguridad y paquetes o extensiones de seguridad de Java (Alarcos, 2013). La idea propuesta es la implementación de una herramienta apta para generar estos requerimientos del cliente/servidor diferenciando de la herramienta IKEYMAN, con la generación de forma automática de la infraestructura de seguridad y el código base necesario para la conexión entre cliente/servidor como se ilustra en la Figura 1.

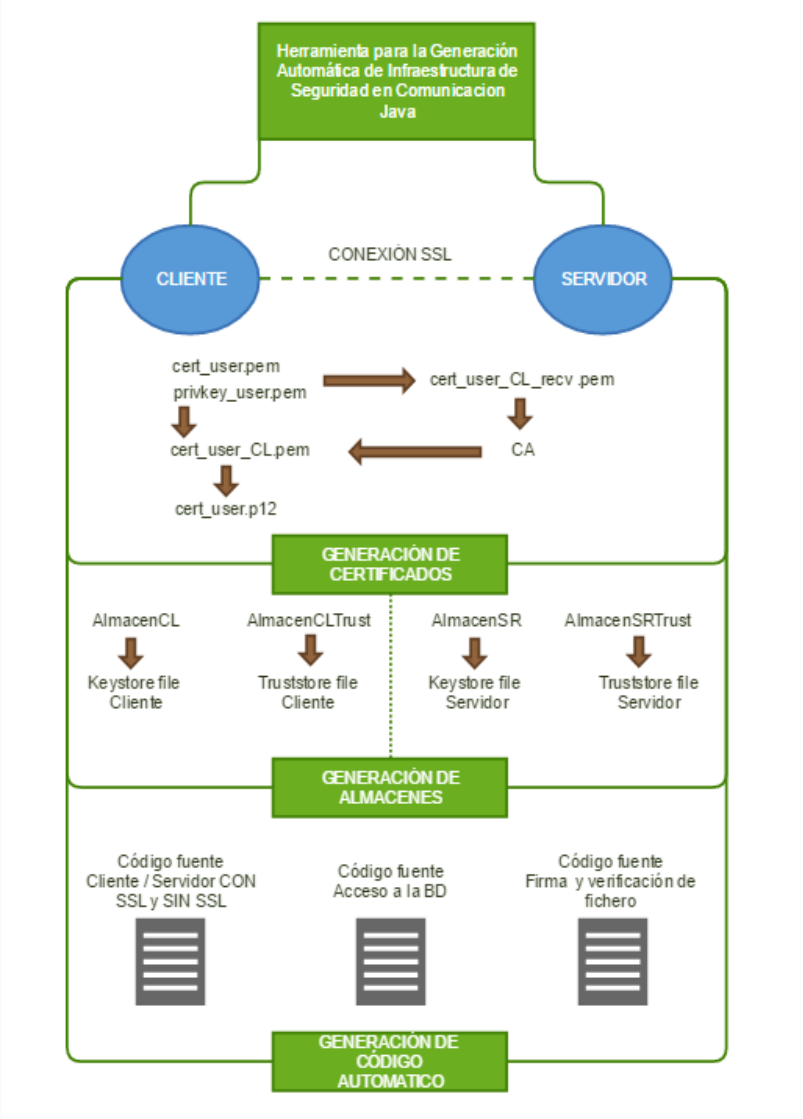

Figura 1. Esquema general de la herramienta.

2. MATERIALES Y MÉTODOS

Este proceso se desarrolla en tres fases de forma automática desde la herramienta. Fase uno consiste en la creación de los certificados, claves públicas y privadas necesarias; fase dos se crean los almacenes de seguridad Java asociados a su aplicación a partir de los certificados y claves mencionadas en la fase uno; fase tres se genera el código fuente base de seguridad asociado a la aplicación de usuario y a los almacenes de la fase dos.

Cada una de las tres fases está asociada a un módulo o sistema específico:

a. **S:A:G:C:** Sistema automático de Generación de Certificados
b. **S:A:G:A:S:** Sistema automático de Generación de Almacenes de Seguridad
c. **S:A:G:C:F:** Sistema automático de Generación de Código Fuente

2.1. Sistema Automático de Generación de Certificados (S:A:G:C)

El desarrollo de una pasarela JNI (Kurzyniec, 2001), para el acceso desde Java a funciones criptográficas de OpenSSL no provistas en la librería JCE de la Máquina Virtual Java (Schaumont, 2003), evita el recurrir a un proveedor externo. De esta manera consigue que desde un proyecto Java se pueda crear un certificado de usuario firmado por la autoridad de certificación para utilizar la infraestructura de gestión de certificados de OpenSSL.

Este sistema permite construir certificados digitales, claves privadas y públicas asociadas que identificando a un usuario o entidad forman parte de la infraestructura básica de seguridad que se va a necesitar a la hora de implementar aplicaciones distribuidas. El sistema firma automáticamente certificados de usuario por una Autoridad de Certificación en adelante CA, la implementación de este módulo es la base del desarrollo de los otros módulos con las siguientes especificaciones:

- conversión de un string writeUTF/readUTF, para la conexión SSL con almacenes de certificados firmados por la CA
- se incorpora la librería OpenSSL; el cliente calcula la longitud del fichero req.pem y se envía al servidor
- se reserva un espacio array de bytes para el fichero req.pem en el cliente para enviar al servidor, luego el servidor recibe el fichero lo almacena en el array de bytes y lo imprime al fichero req_recv.pem
- el servidor firma la petición y genera el fichero cert_user_CL.pem
- el servidor calcula la longitud del fichero cert_user_CL.pem y se la envía al cliente, el cliente reserva espacio en un array de bytes, luego el servidor envía el fichero cert_user_CL.pem al cliente, lo recibe y lo imprime al fichero
- el cliente convierte el fichero cert_user_CL.pem junto con la clave privada a un fichero cert_user.p12

Para que el cliente obtenga los certificados debidamente firmados se crea la carpeta CA en el servidor como se ilustra en la Figura 2, OpenSSL nos proporciona la infraestructura necesaria para crear una propia Autoridad de Certificación, la CA es una entidad capaz de crear y firmar un certificado previa solicitud de un cliente, de esta forma los cliente pueden (si lo consideran oportuno) instalar el certificado de la CA en su Almacén de "entidades emisoras de certificados CA raíz de confianza" (Acebey, 2006).

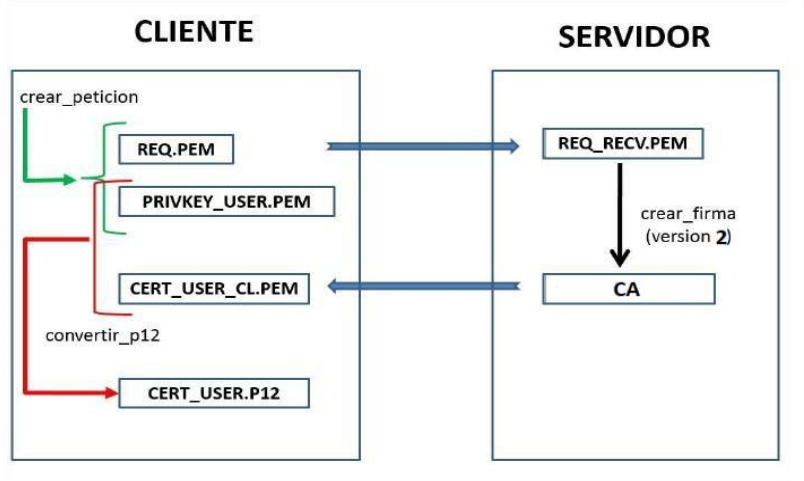

Figura 2. Esquema cliente/servidor con certificados por la CA.Sistema automático de Generación de Almacenes de Seguridad (S:A:G:A:S).

Es un sistema automático que desde Java permite generar almacenes de seguridad necesarios en

aplicaciones de seguridad, bien sea aplicaciones cliente/servidor SSL o aplicaciones de generación y/o verificación de firma, aplicaciones de acceso seguro a una Bases de Datos, etc. Normalmente la generación de almacenes de seguridad se realiza de forma manual con la herramienta Keytool (Ahrendt, 2005), que forma parte de la distribución de la Máquina Virtual Java.

Para el caso de estudio del proyecto se crea estos almacenes de forma automática a partir de los ficheros de certificados por el sistema automático de generación de certificados S:A:G:C, como se ilustra en la Figura 3. El número y la tipología de los almacenes son destinados dependiendo del tipo de entidad o de la aplicación.

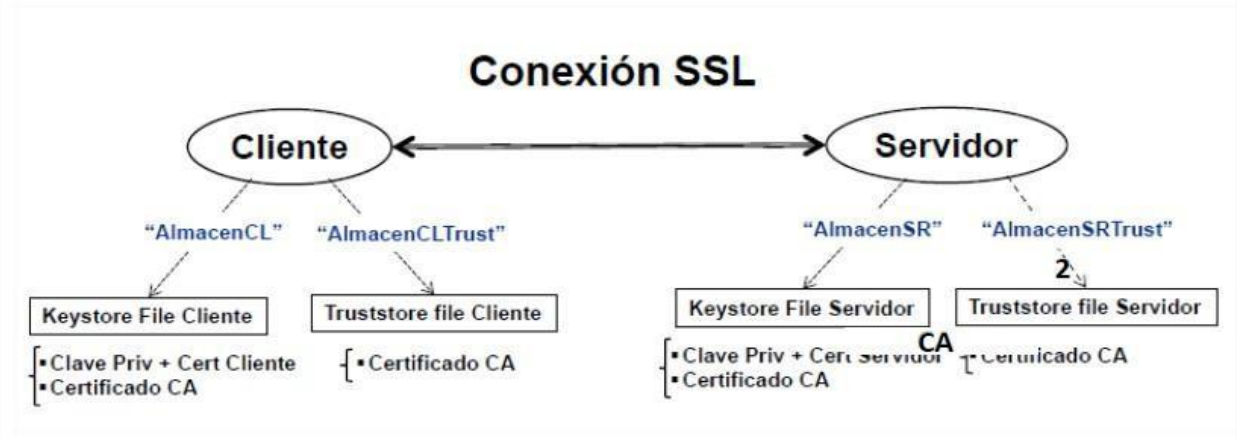

Figura 3. Esquema cliente/servidor con almacenes de certificados.

2.2. Sistema Automático de Generación de Código Fuente (S:A:G:C:F)

Este sistema genera de forma automática la estructura básica del código fuente de las aplicaciones de seguridad del cliente/servidor SSL (Pianegiani, 2003), acceso seguro a Bases de Datos o de aplicaciones de firma y verificación de ficheros. Además, recoge los ficheros generados en el sistema automático de generación de almacenes, integrando con el código fuente generado como se ilustra en la Figura 4, para compilar la aplicación externamente y obtener el código final ejecutable.

```
File coddestino = new File ((System.getProperty("user.dir"))
    + File.separator + "Generacion Codigo" + File.separator + "Cliente_Servidor SIN autenticacion"
    + File.separator + "Cliente" + File.separator + "EchoClient_SA.class");

File coriginalj = new File ((System.getProperty("user.dir"))
    + File.separator + "src" + File.separator + "EchoClient_SA.java");

File cdestinoj = new File ((System.getProperty("user.dir"))
    + File.separator + "Generacion Codigo" + File.separator + "Cliente_Servidor SIN autenticacion"
    + File.separator + "Cliente" + File.separator + "EchoClient_SA.java");

File sdestino = new File ((System.getProperty("user.dir"))
    + File.separator + "Generacion Codigo" + File.separator + "Cliente_Servidor SIN autenticacion"
    + File.separator + "Servidor" + File.separator + "EchoServer_SA.class");

File soriginalj = new File ((System.getProperty("user.dir"))
    + File.separator + "src" + File.separator + "EchoServer_SA.java");

File sdestinoj = new File ((System.getProperty("user.dir"))
    + File.separator + "Generacion Codigo" + File.separator + "Cliente_Servidor SIN autenticacion"
    + File.separator + "Servidor" + File.separator + "EchoServer_SA.java");
```

Figura 4. Código autogenerado dentro de la herramienta.

3. RESULTADOS Y DISCUSIÓN

De acuerdo a las características de la herramienta descrita, se subdivide en etapas o pruebas por cada uno de los módulos: Pruebas (S:A:G:C).

La carpeta o "CA/newcerts", va generando los certificados enviados al cliente, hay que tener en cuenta los detalles de cada certificado enviado como se ilustra en la Figura 5.

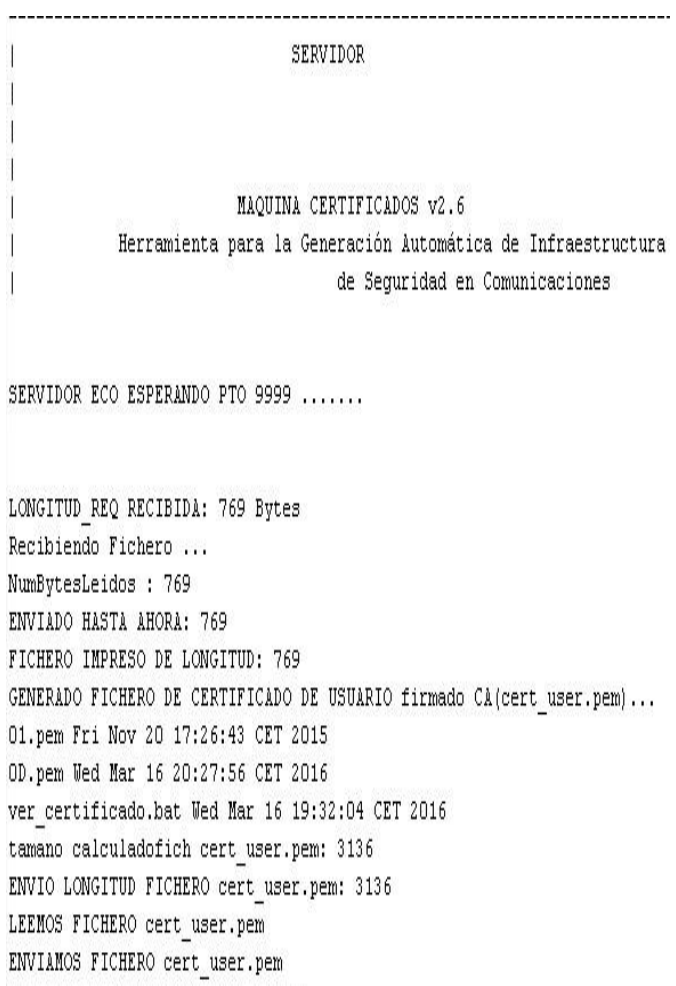

Figura 5. Herramienta, entorno servidor.

La generación de una herramienta cliente/servidor facilita él envió del fichero con el certificado debidamente firmado por la entidad certificadora CA como se ilustra en la Figura 6.

El servidor es el encargado de enviar el fichero firmado por CA para el cliente tenga un certificado validado por el servidor, luego el cliente imprime su certificado debidamente generado como se ilustra en la Figura 7 (Hielscher, 2006).

El cliente SSL carga el fichero AlmacenTrust creado con la herramienta Keytool (Herrera Acebey, 2011), conteniendo el certificado de clave pública del servidor, El cliente verifica el certificado del servidor utilizando el almacén "AlmacenTrust" que contiene el certificado de la CA. Por otro lado, la v2.6 uso de la carpeta CA para generar certificados, utiliza almacenes como AlmacenSRTrust, AlmacenTrust, el servidor verifica este certificado de la misma forma que hacia el cliente con el certificado del servidor ("AlmacenSRTrust") como se ilustra en la Figura 8.

```
|                          CLIENTE
|
|
|
|                    MAQUINA CERTIFICADOS v2.6
|          Herramienta para la Generación Automática de Infraestructura
|                                de Seguridad en Comunicaciones


Direccion IP: 127.0.0.1
Puerto:9999
CREADO SOCKET SSL
CREANDO UNA CSR (CERTIFICATE SIGNING REQUEST):

longitud Clave (1024, 2048, 4096): 1024
common_name: EC
mail_user UTA@GMAIL.COM
countryName: EC
stateOrProvinceName: TU
localityName: AM
organizationName: FISEI
organizationalUnitName: CEDIA
Tamano calculado fich req.pem: 769 Bytes
envio longitud fichero: 769 Bytes
envio  fichero req.pem:
LONG RECIBIDA cert_user.pem..... 3136
Recibiendo Fichero ...
NumBytesLeidos : 3136
ENVIADO HASTA AHORA: 3136
FICHERO IMPRESO DE LONGITUD: 3136
RECIBIMOS FICHERO cert_user.pem.....
IMPRIMIMOS FICHERO cert_user_CL.pem.....
IMPRIMIMOS FICHERO cert_user.p12.....
```

Figura 6. Herramienta, entorno cliente.

```
Z:\compartir\5SEM\MAQUINA_CERTIFICADOS_VERSION_1\MAQUINA_CERTIFICADOS_VERSION_1\
MAQUINA_CERTIFICADOS_v2.6_s>openssl x509 -in cert_user_CL.pem -noout -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 13 (0xd)
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=ES, ST=MA, L=BOA, O=FIM, OU=UPM, CN=CA_DSSR/emailAddress=CA_DS
SR@FI.UPM.ES
        Validity
            Not Before: Mar 16 19:26:36 2016 GMT
            Not After : Mar 16 19:26:36 2017 GMT
        Subject: C=EC, ST=TU, O=FISEI, OU=CEDIA, CN=EC/emailAddress=UTA@GMAIL.CO
M
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (1024 bit)
                Modulus:
                    00:dd:4e:82:50:68:0d:18:3b:bd:4d:79:fa:6a:e7:
                    d9:5e:d0:89:63:ae:f7:b3:52:4f:c5:87:f7:5c:01:
                    c0:84:a6:7b:b9:a8:e5:81:6f:d5:7a:c6:51:62:ff:
                    7a:cd:9a:a2:e0:5d:68:61:03:aa:5a:c9:ce:55:60:
                    b8:d1:b9:36:52:03:76:d5:59:b4:60:25:d2:f7:f3:
                    16:30:c5:5b:35:1b:c2:97:99:3b:73:f3:b2:8b:7f:
                    2a:84:5c:6b:f7:1f:4b:e4:ff:bd:f9:d7:b4:d6:70:
                    75:3e:61:73:26:30:7a:e6:0d:50:45:2a:61:83:1d:
                    57:9c:f8:ce:b9:ca:14:05:63
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                34:4E:98:42:D0:C8:D5:88:75:E4:91:D5:54:DD:DB:C2:F0:68:5A:63
            X509v3 Authority Key Identifier:
                keyid:C8:A3:9B:98:44:3D:EF:CE:50:1A:58:1D:EE:42:C2:A3:53:E2:1F:F
E

    Signature Algorithm: sha1WithRSAEncryption
        8c:d0:c8:e5:e9:ae:45:8e:c8:01:dd:bf:7b:ae:fa:61:03:10:
        4a:c8:f1:8d:07:a1:fb:c7:d9:25:eb:7b:48:fb:a2:a9:7d:1c:
        93:09:d5:cc:06:91:eb:03:97:db:8a:f1:70:54:44:14:05:d9:
        e8:a0:be:d9:5e:21:59:4d:30:be:a0:21:e0:c7:69:57:9f:8a:
        21:6c:cb:71:f3:ec:dd:be:d0:0f:a6:4a:e3:8d:92:c4:96:54:
        f3:da:29:46:73:55:ed:b5:44:d6:66:c9:d4:47:14:83:3c:ef:
        12:25:8d:82:bc:59:c5:4b:b8:51:d8:1b:0d:9f:19:3d:38:93:
        4f:e3

Z:\compartir\5SEM\MAQUINA_CERTIFICADOS_VERSION_1\MAQUINA_CERTIFICADOS_VERSION_1\
MAQUINA_CERTIFICADOS_v2.6_s>pause
Presione una tecla para continuar . . .
```

Figura 7. Comprobación de resultado <<openssl x509 -in cert_user_CL.pem -noout –text>>.

3.1 Pruebas; (S:A:G:A:S)

Figura 8. Resultado de <<keytool -list -v -keystore AlmacenNuevo_25_1652 >>.

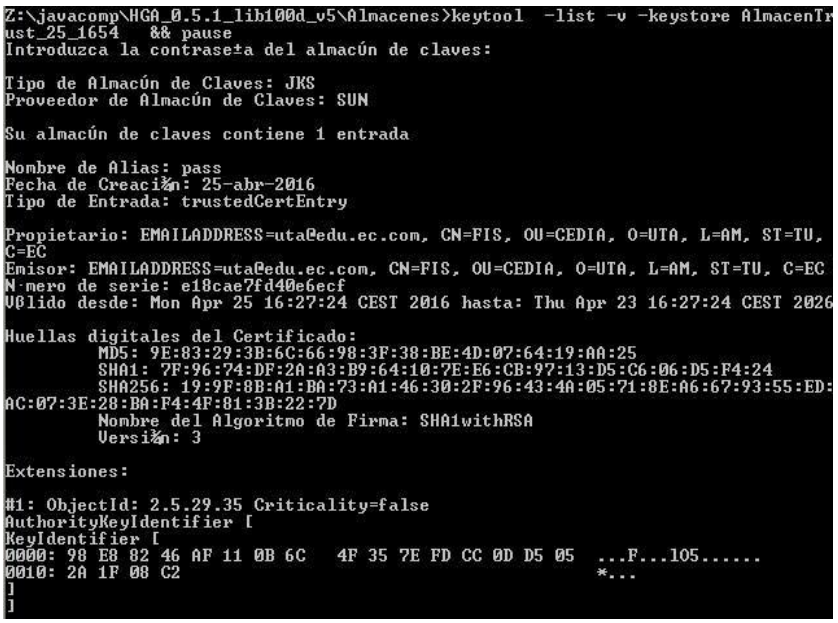

Figura 9. Resultado de <<keytool - list -v -keystore AlmacenTrust_25_1654>>.

Un AlmacenTrust guarda claves criptográficas y certificados (AMRES, 2015), para crear un AlmacenCL con los parámetros tales como el certificado del cliente y la clave privada como se ilustra en la Figura 9.

- El Certificado de la CA en el "AlmacenCLTrust" se almacena;
- Las claves publica y secreta se almacenan en el almacén AlmacenCL;
- Se crea una petición CRS Keytool para firmar del Certificado por una CA;
- El Certificado de la CA en el "AlmacenCL" y firmado por la CA "AlmacenCL";
- "/CA/newcerts", va generando los certificados enviados al cliente, teniendo en cuenta los detalles de cada certificado enviado.

Para el AlmacenTrust, se envía los parámetros a partid del certificado de la CA "CACertificado.pem".

3.2. Pruebas; (S:A:G:C:F)

El cliente SSL carga el fichero AlmacenTrust creado con la herramienta Keytool conteniendo el certificado de clave pública del servidor, generando así el código EchoClient_(SA/AU).java*.class y para el servidor EchoServer_(SA/AU).java*.class.

Figura 10. Prueba; código generado del cliente SIN Autenticación.

El cliente SSL con autenticación establece la conexión con parámetros dentro de los almacenes; AlmacenTrust y AlmacenCL con el certificado firmado por la CA como se ilustra en la Figura 10, por otro lado para autentificar al servidor con parámetros del AlmacenTrust y un certificado del cliente firmado por la CA, se crea partir de la dirección IP del servidor y el puerto de conexión como se ilustra en la Figura 11, con sus respectivos certificados se guardan en la carpeta del servidor nombrada "Cliente Servidor SIN/CON autenticación". Los resultados comprueban la comunicación exitosa entre el cliente y el servidor SIN y CON Autenticación del código generado, además se muestra el almacén del cliente/servidor debidamente firmado.

Los beneficios e impactos del proyecto, realizan un análisis de los usuarios, beneficiarios inmediatos y mediatos, incluyendo a CEDIA y sus miembros para implementar una herramienta que permite generar de forma automática la infraestructura y el código fuente de aplicaciones de seguridad en el desarrollo de aplicaciones propietarias reduciendo el coste de proceso, se consigue potenciar los recursos disponibles para los usuarios de la Red CEDIA, al estar ahora protegidos con servicios de seguridad. El construir de forma sencilla sistemas de transferencia segura de aplicaciones corporativas de firma de documentos por múltiples usuarios reduciendo el coste de la gestión. Además la renovación temporal de la infraestructura de seguridad es fácilmente actualizada con la herramienta desarrollada: Es decir, cuando caduquen los certificados digitales y/o los almacenes de seguridad la herramienta recreara la nueva infraestructura de forma automática.

Figura 11. Prueba; código generado del servidor SIN Autenticación.

4. CONCLUSIONES

La Herramienta para la Generación Automática de Infraestructura de Seguridad en Comunicaciones usando Java, facilita los certificados de forma automatizada con el uso de almacenes o base de datos de claves criptográficas para entidades públicas como privadas dentro de la Red CEDIA, los certificados son firmados por una Autoridad de Certificación para un tiempo determinado. La idea de verificar y validar un certificado dentro de la herramienta minimiza el coste de tiempo de su creación de forma manual.

El proceso de crear un almacén de certificados y claves para cada entidad, facilita a los usuarios que de forma automática se generen nuevos certificados debidamente firmados; una vez que estos caduquen, previamente al importar un nuevo certificado de una CA necesita uno o más certificados de confianza en el almacén de claves, por lo que la herramienta crea un código base que facilita esta transacción entre cliente/servidor.

La generación de código base, provee un código descifrable para el usuario que le permite realizar una consulta a los almacenes, accediendo a las aplicaciones de seguridad entre cliente/servidor con autenticación y sin autenticación, Base de Datos seguras o de aplicaciones de firma y verificación de documentos, implicado en una nueva infraestructura de forma automática con la herramienta que logró automatizar correctamente el proceso de creación de certificados, utilizando una librera OpenSSL implantada.

AGRADECIMIENTO

Agradecimientos: Al Consorcio Ecuatoriano para el Desarrollo de Internet Avanzado CEDIA, por el financiamiento brindado a la investigación, desarrollo e innovación, mediante los proyectos CEPRA, en especial al proyecto CEPRA-IX-2015; Herramienta para la Generación Automática de Infraestructura de Seguridad en Comunicaciones usando Java.

REFERENCIAS

Acebey, J.H., 2006. Certificados digitales. *Revista Acta Nova*, 3(3), 1-11.

Ahrendt, W., T. Baar, B. Beckert, R. Bubel, M. Giese, R. Hähnle, P.H. Schmitt, 2005. The key tool. *Software & Systems Modeling*, 4(1), 32-54.

Alarcos, B., E.D.L. Hoz, M. Sedano, M. Calderón, 2003. *Performance analysis of a security architecture for active networks in Java*. Disponible en http://www.google.com.ec/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=0ahUKEwjmqrzvka7PAhXH7D4KHaSHDIIQFggiMAE&url=http%3A%2F%2Fciteseerx.ist.psu.edu%2Fviewdoc%2Fdownload%3Fdoi%3D10.1.1.579.4621%26rep%3Drep1%26type%3Dpdf&usg=AFQjCNFxY_7CYFeO7H08aNCnwlBbXJGpxg&bvm=bv.133700528,d.cWw.

Clark, J., P.C. van Oorschot, 2013. SoK: *SSL and HTTPS: Revisiting past challenges and evaluating certificate trust model enhancements.* In Security and Privacy (SP) 2013 IEEE Symposium on, pp. 511-525.

Gröbert, F., 2010. *Automatic identification of cryptographic primitives in software*. Deplima Thesis, Ruhr-University Bochum, Germany.

Hielscher, R.P., V. Delgado, 2006. Aplicaciones prácticas de la criptografía. *Anales de Mecánica y Electricidad*, 83(2), 10-16.

Khalil-Hani, M., V.P. Nambiar, M.N. Marsono, 2010. *Hardware Acceleration of OpenSSL cryptographic functions for high-performance Internet Security*. In: Intelligent Systems, Modelling and Simulation (ISMS), 2010 International Conference on, pp. 374-379.

Kurzyniec, D., V. Sunderam, 2001. *Efficient cooperation between Java and native codes–JNI performance benchmark.* In: The 2001 international conference on parallel and distributed processing techniques and applications. Disponible en https://www.researchgate.net/publication/228752983_Efficient_cooperation_between_Java_and_native_codes-JNI_performance_benchmark.

Lakhe, B., 2014. *Setting Up a KeyStore and TrustStore for HTTP Encryption*. In: Practical Hadoop Security (pp. 181-182), Apress.

López Jiménez, J.M., J.F. Otria Silva, E.P. Santiago Posadas, 2015. *Implementación de una autoridad certificadora con la herramienta openca para generar certificados digitales*. Disponible en http://tesis.ipn.mx/handle/123456789/15042.

Pajin, M.K.A., D., Bukvić, M. Stojaković, I. Barišić, B. Jakovljević, 2015. *Securing service access with digital certificates*. Disponible en http://services.geant.net/cbp/Knowledge_Base/Security/Documents/gn3-na3-t4-abpd106.pdf, 58 pp.

Park, H., S. Redford, 2007. *Client certificate and IP address based multi-factor authentication for J2EE web applications*. In: Proceedings of the 2007 conference of the center for advanced studies on Collaborative research, pp. 167-174. IBM Corp.

Pianegiani, F., D. Macii, P. Carbone, 2003. An open distributed measurement system based on an abstract client-server architecture. *Instrumentation and Measurement, IEEE Transactions on*, 52(3), 686-692.

Schaumont, P., I. Verbauwhede, 2003. Domain-specific codesign for embedded security. *Computer*, 36(4), 68-74.